02

Viendo las modificaciones

Transcripción

[00:00] Hola todos y todas. Bienvenidos y bienvenidas una vez más a este curso de Git. Ya tenemos un proyecto finalizado, hicimos todas las modificaciones necesarias en el título, todos los nombres de los cursos están correctos y ya vimos cómo trabajar en equipo con repositorios remotos, con branches separadas, corrigiendo conflictos, trayendo datos de una branch para otra, actualizando una branch.

[00:24] En fin, ya vimos varias cosas. Pero supongamos que queremos ver qué ocurrió en cada commit para garantizar que no se agregó ningún bug y para saber qué generamos en cada commit. ¿Cómo puedo ver qué agregamos en cada commit y la diferencia entre un commit y otro?

[00:43] Por ejemplo vamos a loguearnos como Bruno y sabemos que con git log -p obtenemos acá un detalle bastante grande de cada commit. Y por ejemplo ahora apretando Q, utilizando un git log --oneline, vemos por ejemplo cada uno de los commits que hemos hecho. Y si yo quisiera ver la diferencia desde el momento en el que trajimos la lista, por ejemplo esta branch de lista y el commit actual, hasta ahora.

[01:29] Todo lo que fue modificado, pero quiero ver todo eso de una sola vez. ¿Cómo puedo hacer? Existe un comando bien interesante y poderoso de Git llamado git diff. Entonces, este git diff muestra la diferencia entre códigos, solo que si lo pongo solo en la terminal, por ejemplo pongo ahora git diff, enter, no va a pasar nada. ¿Por qué? Por ahora no hay ninguna modificación hecha que no se haya guardado en un commit.

[02:00] Entonces quiero mostrar la diferencia entre dos commits. Para eso tengo que informar dos commits. Vamos a agarrar el hash del merge acá con la lista, vamos a copiar y vamos a hacer git difícil, pegar, y ahora hasta, o sea queremos poner hasta desde es commit hasta el actual. Vamos a copiar acá este. Ese hasta se simboliza con esos dos puntos.

[02:32] Entonces commit, primer commit, dos puntos, segundo commit, clic derecho copiar, pegar, enter y vean lo que está ocurriendo acá. Nos está diciendo qué cosas se modificaron, cuáles de los cursos modificamos, tanto el de Vagrant, Ansible, Kubernetes.

[02:55] Además de eso, por ejemplo si yo estoy modificando alguna cosa, vamos a hacer por ejemplo que estoy acá agregando una línea nueva para un nuevo curso y supongamos que tuve que parar porque es la hora del almuerzo o necesité ir a una reunión y ya había hecho modificaciones de un archivo, ahora apreto "Ctrl + S" por ejemplo, y quiero ver todo lo que he modificado hasta entonces.

[03:19] Puedo simplemente ir a la terminal y hacer un git diff, enter, y ahí vemos por ejemplo qué cosas he modificado pero que aún no he agregado para hacer un commit. Por ejemplo, si ahora yo agrego este archivo con un git add index.html, lo agrego, vamos a ver un git status ahora.

[03:45] Vean que están agregados, si yo hago un git diff, miren, no pasa nada, o sea no aparece nada. ¿Por qué? Porque yo ya lo agregué a los commits. Entonces ese git diff solo funciona para las cosas que no han sido agregados al stage. Si yo quiero, por ejemplo vamos a volver atrás este archivo, porque esa modificación que hice no es relevante en realidad. Entonces, vamos a hacer el git restore --stage index.html.

[04:19] Pongo git status y ahora simplemente hacemos un git restore index.html. Enter. Perfecto. Ahora volvemos a tener nuestro archivo como estaba antes. Entonces, así podemos comenzar a analizar con más control

todas las modificaciones que fueron agregadas durante un desarrollo de un proyecto. Conseguimos tener un control más fino entre ellos, ver las modificaciones entre commits, entre lo que estamos haciendo ahora y lo que ya fue commiteado.

[04:53] Entonces con git diff podemos ver todas las modificaciones, pero después de haber visto todas la modificaciones y garantizado que no hay bugs, ¿cómo puedo generar una versión? Por ejemplo una release o la versión 0.1. ¿Cómo puedo informar esto para Git, clavar una bandera acá y decir: "ese commit de acá es la release 0.1"? Bueno, vamos a ver eso en el próximo video.